PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	10/750,517	Confirmation No. 9240
Applicants	:	Michael G. Lisanke et al.	
Filed	:	December 31, 2003	
TC/A.U.	:	2139	
Examiner	:	Harris C. Wang	
Docket No.	:	SOM920030007US1	
Customer No.	:	78007	

APPEAL BRIEF

MS-APPEAL BRIEF-PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is filed in response to a Final Office Action dated May 12, 2008, followed by a Notice of Appeal with Pre-Appeal Conference Request Brief filed November 11, 2008, and a Notice of Panel Decision dated December 15, 2008. Reconsideration of the Application, withdrawal of the rejections, and allowance of the claims are respectfully requested.

CERTIFICATE OF TRANSMISSION

In accordance with 37 CFR 1.8, I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 or facsimile transmitted or submitted under electronics filing system to the U.S. Patent and Trademark Office on the date:

January 15, 2009 By: Jon A. Gibbons Signature: /Jon A. Gibbons/

(Applicant, Assignee, or Representative)

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines (IBM) of Armonk, NY.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-22 are pending.

Claims 1-22 are rejected.

The Appellant is appealing the rejection of independent claims 1, 10, 15, and 20 (and all other remaining claims that depend from these claims). Claims 1, 10, 15, and 20 are on appeal.

IV. STATUS OF AMENDMENTS

The Examiner issued a final rejection of claims 1-22 in the Final Office Action of May 12, 2008. Appellants submitted a Notice of Appeal with Pre-Appeal Conference Request Brief filed November 11, 2008 in response to the Final Office Action.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

This summary references page numbers and paragraph numbers of the Pre-Grant Publication No. 2005/0149750 that corresponds to the Specification as originally filed.

The pending independent claims under appeal in this case are corresponding system, methods, and computer readable storage medium claims. The following identifies the subject matter defined in each of the claims under appeal in the present application.

Independent system Claim 1 sets forth the following subject matter.

A) a processor which monitors at least one instance of software execution,: page 2, paragraph [0016] and page 3, paragraph [0041].

B) wherein the one instance is identified and selected by an end-user to be monitored by the processor,: FIG. 8 showing the user detects problem 200 and start secure logger 220; page 2, paragraphs [0016], [0017], [0018] and page 5, paragraphs [0050] and [0051].

C) wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, and: FIG. 8 showing the user detects problem 200 and start secure logger 220 and page 5, paragraphs [0050], and [0051].

D) the wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored,: FIG. 2 showing Logger Initializer 74, Logger Runtime 76, and Logfile Viewer 78; and FIG. 8 showing Log Messages 230 and Log File Sent 245; and page 2, paragraphs [0017], [0018]; and page 5, paragraph [0051].

E) whereby the set of data is used to diagnose the software execution;: FIG. 8 showing Problem Analysis 255; and page 1, paragraph [0012]; page 2, paragraph [0016]; and pages 3-4, paragraph [0041].

F) an encryption system which generates at least one symmetric key and encrypts the log entry for the at least one set of data using the symmetric key,: FIG. 3 showing build keys 82; Abstract; page 2, paragraphs [0017], [0018], and [0020]; pages 3-4, paragraph [0041]; and page 4, paragraphs [0046] and [0047].

G) wherein the encryption system encrypts the symmetric key using a public key associated with the encryption system;: page 2, paragraph [0020].

H) a log file of a relatively-fixed size which stores the log entry for the at least one set of data which has been encrypted,: all of FIGs. 7A, 7B, and 7C; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047] , and [0049].

I) and wherein the log file includes the symmetric key which has been encrypted with the public key;: pages 3-4, paragraph [0041].

J) random data in the log file when it is originally created and which is replaced by log entries so that a size of the log file including log entries appears to be a substantially constant size; and: pages 3-4, paragraph [0041] and page 4, paragraphs [0045]- [0047].

K) a pointer which identifies a next storage location for a next log entry so that a last log entry can be determined and the next log entry can be positioned in a location in the log file after a previous log entry.: page 4, paragraph [0048].

Independent method Claim 10 sets forth the following subject matter.

A) generating a log file full of random data, wherein the log file is of a substantially constant size;: all of FIGs. 7A, 7B, and 7C; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047], and [0049].

B) generating at least one symmetric key, wherein the symmetric key resides within the log file;: FIG. 3 showing build keys 82; Abstract; page 2, paragraphs [0017], [0018], and [0020]; pages 3-4, paragraph [0041]; and page 4, paragraphs [0046] and [0047].

C) turning on logging and establishing a pointer for a location of a next logged software operation activity;: FIG. 2 showing Logger Initializer 74, Logger Runtime 76, and Logfile Viewer 78; and FIG. 8 showing Log Messages 230 and Log File Sent 245; and page 2, paragraphs [0017], [0018], page 4, paragraph [0048]; and page 5, paragraph [0051].

D) monitoring at least one software operation activity within the tamper-resistant environment and generating messages in response to at least one instance of software execution within the tamper-resistant environment, wherein the software operation activity is identified and selected

by an end-user to be monitored,: page 2, paragraphs [0016 and [0019]; and pages 3-4, paragraphs [0041], [0046]-[0047], and [0049]; and page 5, paragraph [0051].

E) wherein the end-user is a user that initiates execution of the software at a system associated with the end-user;: FIG. 8 showing the user detects problem 200 and start secure logger 220 and page 5, paragraphs [0050], and [0051].

F) encrypting, in response to the monitoring, a record associated with each generated message using the symmetric key;: FIG. 3 showing build keys 82; Abstract; page 2, paragraphs [0017], [0018], and [0020]; and pages 3-4, paragraph [0041]; and page 4, paragraphs [0046] and [0047].

G) encrypting using the symmetric key, wherein [[the]]a encryption system encrypts the symmetric key using a public key associated with the encryption system;: page 2, paragraph [0020].

H) logging into a log entry into a log file at least one software operation activity relating to a generated message by replacing random data with an encrypted record of the software operation activity;: all of FIGs. 7A, 7B, and 7C; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047] , and [0049].

I) placing into the log file the symmetric key which has been encrypted with the public key;: pages 3-4, paragraph [0041].

J) moving the pointer when the log entry has been made to a next available log position;: page 4, paragraph [0048].

K) wrapping the pointer to a beginning of the log file when the log file is full of log entries;: page 2, paragraph [0032]; pages 3-4, paragraphs [0041], [0043], and [0047]-[0048].

L) sending the log file to a secure location where the log file can be decrypted and analyzed; and: FIG. 8 showing Log File Sent 245; page 4, paragraph [0043] and page 5, paragraph [0051].

M) analyzing decrypted log file data and providing information for diagnosing software in the tamper-resistant environment.: FIG. 8 showing Problem analysis 255; page 4, paragraphs [0043] and [0049]; and page 5, paragraphs, [0051] and [0052].

Independent method Claim 15 sets forth the following subject matter.

A) receiving from the remote protected processing environment an encrypted log file of substantially-constant size comprising at least one log entry with at least one set of data derived from at least one instance of software execution monitored in response to an end-user identifying and selecting the one instance of software execution,: all of FIGs. 7A, 7B, and 7C and FIG. 8 showing the user detects problem 200 and start secure logger 220; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047] , and [0049]; page 5, paragraphs [0050], and [0051].

B) wherein the end-user is a user that initiates execution of the software at a system associated with the end-user,: FIG. 8 showing the user detects problem 200 and start secure logger 220 and page 5, paragraphs [0050], and [0051].

C) whereby the set of data is used to diagnose the software execution, wherein the set of data within the encrypted log file has been encrypted with at least one symmetric key included within the encrypted log file,: FIG. 8 showing Problem analysis 255; page 2, paragraphs [0017], [0018], and [0020]; page 3-4, paragraphs [0041]-[0043], [0046], [0047], and [0049]; and page 5, paragraphs, [0051] and [0052].

D) and wherein the symmetric key has been encrypted by a public key associated with an encryption system;: page 2, paragraph [0020].

E) determining a decrypting key for the encrypted log file and decrypting the encrypted log file;: FIG. 6 showing Decrypt 112 and FIG. 8 showing Decrypt Log File 250; page 4, paragraph [0043] and page 5, paragraph [0051].

F) determining a private decrypting key corresponding to the public key associated with the system;: page 2, paragraph [0020] and page 3, paragraph [0040].

G) analyzing, using the decrypting key and the private decrypting key, the log entry at the remote protected processing environment to determine whether an operation of the remote protected processing environment corresponding to the at least one set of data derived from at least one instance of software execution is appropriate; and: FIG. 8 showing Problem analysis 255; page 4, paragraphs [0043] and [0049]; and page 5, paragraphs, [0051] and [0052].

H) reporting results of the analyzing step.: FIG. 8 showing Report To User 260; page 5, paragraph [0051].

Independent computer readable storage medium Claim 20 sets forth the following subject matter.

A) recording at least one set of data serviced from at least one instance of software execution identified and selected by an end- user to be monitored whereby the set of data is used to diagnose the software execution wherein the end-user is a user that initiates execution of the software at a system associated with the end-user;: all of FIGs. 7A, 7B, and 7C and FIG. 8 showing the user detects problem 200 and start secure logger 220; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047] , and [0049]; page 5, paragraphs [0050], and [0051].

B) generating at least one symmetric key;: page 2, paragraph [0020] and pages 3-4, paragraph [0041].

C) encrypting the symmetric key using a public key associated a client computer system: page 2, paragraph [0020].

D) encrypting the recording of the at least one set of data using the symmetric key;: FIG. 3 showing build keys 82; Abstract; page 2, paragraphs [0017], [0018], and [0020]; and pages 3-4, paragraph [0041]; and page 4, paragraphs [0046] and [0047].

E) recording the at least one set of data, which has been encrypted sequentially in a storage block of a substantially fixed size,: all of FIGs. 7A, 7B, and 7C; and page 2, paragraphs [0017]-[0021]; pages 3-4, paragraphs [0039]-[0042], [0045]-[0047] , and [0049].

F) wherein the storage block includes the symmetric key which has been encrypted with the public key;: pages 3-4, paragraph [0041].

G) maintaining a pointer to a next available location for recording the at least one set of data sequentially in the storage block;: page 4, paragraph [0048].

H) responding to a command and sending an encrypted log file comprising the at least one set of data which has been encrypted and sequentially recoded in the storage block to a remote location for decryption and analysis. FIG. 8 showing Problem analysis 255 and showing Report To User 260; page 4, paragraphs [0043] and [0049]; and page 5, paragraphs, [0051] and [0052].

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Whether claims 1-22 fail to comply with the written description requirement.

B. Whether claims 1-22 are unpatentable under 35 U.S.C. §103(a) over *Circenis* (U.S. Pre-Grant Publication No. 2004/0054908) in view of the *OpenPGP* standard (RFC 2440) further in view of *The IBM Certification Study Guide AIX V4.5 System Administration (1999).*

VII. ARGUMENT

A. WHETHER CLAIMS 1-22 FAIL TO COMPLY WITH THE WRITTEN DESCRIPTION REQUIREMENT

In the Examiner's Office Action of May 12, 2008, the Examiner rejected claims 1-22 under 35 U.S.C. §112, first paragraph, stating that independent claims 1, 10, 15, and 20 contain new subject matter. The Appellant respectfully disagrees with the Examiner.

The following chart lists the language that the Examiner states is new matter and also identifies examples where support for this language can be found in the U.S. Pre-Grant Publication No. 2005/0149750 of the Specification as originally filed.

(1) wherein the one instance is identified and selected by an end-user to be monitored by the processor	FIG. 8 and paragraphs [0016], [0017], [0018], [0050], and [0051]
(2) wherein the end-user is a user that initiates execution of the software at a system associated with the end-user,	FIG. 8 and paragraphs [0050], and [0051]
(3) and wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored…	FIG. 2 and FIG. 8 and paragraphs [0017], [0018], and [0051]

As can be seen from the above chart, the Specification as originally filed comprises sufficient support for the claim language added in the Response With Amendment dated January 2, 2008. The Examiner argues that because the Specification states that an advantage of the present invention is that the "content of messages logged are all obscured from the user of the system on which the logging is done…it is more difficult for the user to determine what is being logged" that a user would be unable to select an instance of software execution to be monitored. However, this argument is misplaced. For example, the Specification as originally filed does not state that a user cannot select an instance of execution to be monitored, but that the user would have a difficult time finding the log and identifying the data in the log. See paragraph [0019] of the Specification as originally filed. Paragraphs [0050] and [0051] clearly show that a user can select an instance of software execution and have that instance monitored. Accordingly, Appellants respectfully suggest that the rejection of claims 1-22 under 35 U.S.C. §112, first paragraph, has been overcome and should be withdrawn.

B. WHETHER CLAIMS 1-22 ARE UNPATENTABLE UNDER 35 U.S.C. §103(A) OVER *CIRCENIS* (U.S. PRE-GRANT PUBLICATION NO. 2004/0054908) IN VIEW OF THE *OpenPGP* STANDARD (RFC 2440) FURTHER IN VIEW OF *THE IBM CERTIFICATION STUDY GUIDE AIX V4.5 SYSTEM ADMINISTRATION (1999)*

In the Examiner's Office Action of May 12, 2008, the Examiner rejected claims 1-22 under 35 U.S.C. §103(a) as being unpatentable over *Circenis* (U.S. Pre-Grant Publication No. 2004/0054908) in view of the *OpenPGP standard (RFC 2440)* further in view of *The IBM Certification Study Guide AIX V4.5 System Administration (1999)*, herein after referred to as the "*IBM AIX*". The Appellant asserts that neither the *Circenis*, the *OpenPGP standard*, or *IBM AIX* references, taken either alone or in combination with one another, teach or suggest the claimed limitations.

The Appellant respectfully suggests selection of independent Claim 1 as representative of the independent claims on appeal. Independent Claim 1 is directed towards a system comprising:

> a processor which monitors at least one instance of software execution, wherein the one instance is identified and selected by an end-user to be monitored by the processor, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, and wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored, whereby the set of data is used to diagnose the software execution;
> an encryption system which generates at least one symmetric key and encrypts the log entry for the at least one set of data using the symmetric key, wherein the encryption system encrypts the symmetric key using a public key associated with the encryption system;
> a log file of a relatively-fixed size which stores the log entry for the at least one set of data which has been encrypted, and wherein the log file includes the symmetric key which has been encrypted with the public key;
> random data in the log file when it is originally created and which is replaced by log entries so that a size of the log file including log entries appears to be a substantially constant size; and
> a pointer which identifies a next storage location for a next log entry so that a last log entry can be determined and the next log entry can be positioned in a location in the log file after a previous log entry.

The Appellant asserts that, in particular, the underlined portions of the above claims are not taught or suggested by the *Circenis,* the *OpenPGP standard*, and/or *IBM AIX* references, taken either individually or in combination with one another.

The claims were rejected under 35 U.S.C. §103(a). The Statute expressly requires that obviousness or non-obviousness be determined for the claimed subject matter "as a whole," and the key to proper determination of the differences between the prior art and the presently claimed invention is giving full recognition to the invention "as a whole." As discussed below, the Appellant asserts that these limitations, especially when considered in the context of the other limitations of Claim 1, are not described in the prior art references of record and that these limitations render the claimed subject matter non-obvious over the prior art.

The presently claimed invention is advantageous over the prior art for many reasons. For example, the presently claimed invention provides a simple, yet effective, way of providing a protected operating environment while allowing for capture of information for diagnosis or analysis of program operation to discern problems with the software. The presently claimed invention also has the advantage of avoiding (or minimizing the risk of) compromising the security of the system that includes the tamper resistance built into that system, through either hardware or software which accomplishes the tamper resistance. The presently claimed invention has the advantage that the operation of software within a protected processing environment may be reviewed and errors detected and corrected. This error detection and correction may occur without disabling or otherwise removing or circumventing the protection of the protected processing environment. This is desirable since the protected processing environment may be a source of the problem and removing it would change the operating system and possibly remove the source of the errors.

Another advantage is that the results of monitoring may be selectively engaged, when a problem has occurred or is possible to occur. In this way the program which records the processing takes system resources only when diagnostic information is desired and may be turned "off" or disengaged when no monitoring is needed. Therefore, the processing of the diagnostic routines do not require processing steps when there is no need for the monitoring.

An additional advantage is that the presently claimed invention creates a trace log which is encrypted (preferably as each entry in the log is created) and of fixed size (either relatively or in absolute terms) to protect it from monitoring and the system as a whole from attack. That is, the trace log is file of a fixed size, and the fixed size reduces the chance that its role as a trace log will be apparent from a monitoring of the log and that the events being monitored will be apparent to an attacker, making the trace log and the system as a whole harder to attack. The encryption prevents the contents of the file (trace log) from being read without a decryption key.

A trace operation can be selectively engaged when a log is desired. The inputs and outputs of the system can then be recorded on each activity in a trace log that is encrypted as it is created. The recorded trace log is then sent to a central location where the decryption key is present and the activities in the trace log are decrypted, then analyzed to determine whether each is proper and determining if there was an anomaly and what caused the anomaly.

The presently claimed invention also has the advantage that the times of message logging, the amounts of data logged and the content of messages logged are all obscured from the user of the system on which the logging is done. This has the advantage that the user may know that the system has a logging function, but it is more difficult for the user to find the logging and to determine what is being logged. Further, the messages are obscured by encryption to make the content of the logged messages unintelligible without a decrypting key.

Overview of Prior Art

The *Circenis* reference is directed towards a tamper-evident data management system. The tamper-evident data management system of *Circenis* uses public-private digital signature keys to control use of data and to ensure the fidelity of data that is stored on a customer's system for later collection by a computer vendor or that is sent to the vendor over a network. An application collects usage or metrics data from the computer system. The metering application

uses an application private key to digitally sign all metrics data prior to optionally storing the data in a data log file. The vendor can then use an application public key to validate the digitally signed entries. The digitally signed data entries may also be encrypted using a vendor public key prior to storage in the data log and may be decrypted using a vendor private key prior to validating the digitally signed data. The application and application private key may be stored on a smart card to discourage and detect tampering or may be stored on the computer system itself. See *Circenis* at the Abstract and generally.

The *OpenPGP standard (RFC 2440)* provides specifications for an Internet standards track protocol for the Internet community.

The *IBM AIX* reference provides a study guide for the AIX system.

Cited References Fail to Describe All Limitations

The Examiner in the Final Office Action states that *Circenis* teaches:

> a processor which monitors at least one instance of software execution, wherein the one instance is identified and selected by an end-user to be monitored by the processor, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user is a user that initiates execution of the software at a system associated with the end-user and wherein the processor creates a log entry with at least one of a set of data is used to diagnose the software execution in response to the one instance being identified and selected to be monitored; *("Using the tamper-evident system 200 of Fig. 3, a sender is able to monitor and control application utilization by collecting data associated with the application, creating tamper-evident data records, and providing the tamper-evident data records" Paragraph [0037])*

The Appellant respectfully disagrees. For example, nowhere does Circenis teach or suggest that the end-user selects a particular instance of software execution for monitoring. In fact, the entire focus of *Circenis* is to detect end-user tampering of usage data. As expressly taught by *Circenis*, the sender is the **data owner** (See *Circenis*, for example, at paragraph [0018]), whereas a user in the presently claimed invention is an end user or an IT professional. In other words, *Circenis* is directed at monitoring content usage and tampering of content control policies (See *Circenis* generally), while the present invention is directed at debugging an

application running within a protected environment (See the Specification as originally filed at, for example, pages 3, 7, and 14). The customer of *Circenis* is more comparable to the user of the presently claimed invention, where the customer of *Circenis* is actually using the application, music file, etc.

The **data owner** of *Circenis* configures an application to monitor every instance of an application use, CPU operation, or whatever data is being collected. (See *Circenis* at, for example, the Abstract and paragraphs [0018] and [0019]. The presently claimed invention, on the other, hand, is only monitoring a specific instance of a software execution **that has been selected by the end-user**. The Examiner argues that because the processor of *Circenis* is monitoring every instance of execution that when an end-user executes the software on a device this constitutes the end-user identifying the instance of execution to be monitored. However, this argument inappropriately broadens the scope of *Circenis* in view of the teachings of *Circenis*. The independent claims explicitly recite "wherein the one instance is identified and selected by an end-user to be monitored by the processor, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, and wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored..." In other words, the end-user of the presently claimed invention is explicitly identifying and selecting an instance of software execution to be monitored by the processor.

In contrast, *Circenis* teaches that the data owner is the one who identifies and that **every instance** of application execution is to be monitored and **not** the end-user. *Circenis* only teaches that customer executes an application. Nowhere does *Circenis* teach or suggest that the customer selects an instance of execution for monitoring. In fact, *Circenis teaches away* from this claim element. The entire purpose of *Circenis* is to detect any tampering of a file by a user. Therefore, if a user was given the ability select an instance of software execution to be monitored, as recited for the presently claimed invention, the tamper-evident management system in *Circenis* would be defeated. Because *Circenis* explicitly teaches that a data owner configures an application to monitor each execution of an application and that a customer merely uses an application, the Examiner's argument that the processor of *Circenis* is monitoring every instance of execution

that when an end-user executes the software on a device this constitutes the end-user identifying the instance of execution to be monitored is improper. Accordingly, the presently claimed invention distinguishes over *Circenis* for at least these reasons.

Furthermore, the monitoring of the presently claimed invention is performed on the instances selected by the end-user. *Circenis* is completely silent on this claim element. For example, *Circenis* teaches a metering application that collects "metrics data associated with operation of the computer system" whenever a user uses an application. See *Circenis* at paragraph [0019]. If a user uses an application 5 times, *Circenis* teaches that usage information for each of the 5 times is recorded. Assuming arguendo that *Circenis* and the presently claimed invention teaches logging and monitoring the same type of data (which they do not), *Circenis* would have to teach that of the 5 times an application is used a user can select which of the 5 times data should be logged. *Circenis* clearly does not teach this. In fact, this is completely against what *Circenis* is trying to accomplish as stated above. The Examiner states in the Response to Arguments section of the Final Office Action that the claim language does not state that the monitoring is only performed on the instances selected by the end-user. However, the claim language does explicitly state that the monitoring is performed on the instance identified and selected by the end-user to be monitored i.e., "wherein the one instance is identified and selected by an end-user to be monitored by the processor, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, and wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored…." This is different from monitoring each application user as configured by a data owner, which is taught by *Circenis*. The claim language differs from arbitrarily monitoring every instance of software execution by reciting that the instance identified and selected by a user to be monitored is the instance that is monitored by the processor. Accordingly, the presently claimed invention distinguishes over *Circenis* for at least these reasons as well.

Additionally, claim 1 also recites "…whereby the set of data is used to diagnose the software execution…" The Appellant is unsure of how the Examiner concluded that *Circenis* teaches this element. *Circenis* records metrics for pay-per-use data and/or DRM usage data.

Nowhere does *Circenis* teach that this data is used to diagnose the software execution. Accordingly, the presently claimed invention distinguishes over *Circenis* for at least this reason as well.

On page 8 of the Final Office Action, the Examiner correctly states that *Circenis* and the *OpenPGP* standard do not explicitly teach "random data in the log file when it is originally created and which is replaced by log entries so that a size of the log including log entries appears to by a substantially-constant size". The Examiner goes on to state "It would have been obvious to one or ordinary skill in the art at the time of the invention to modify the monitoring system of Circenis to store the encrypted log entries in a circular log as described by IBM."

The Appellant respectfully disagrees with the Examiner's interpretation of *IBM AIX*. As discussed extensively throughout the prosecution history, it is not customary to insert random data into log files. The Examiner in the Response to Arguments section of the Final Office Action states "The Examiner never argues that it is inherent that the circular file is always populated with random data when created, only that the file is populated with some data when created. Whenever a file is created, of any size, some value is inside the file. Even in a blank entry, some value exists inside the file." However, the presently claimed invention does not recite that the log file is merely populated with some data, but is populated with "random data in the log file when it is originally created and which is replaced by log entries so that a size of the log file including log entries appears to be a substantially constant size". Merely populating a file with some data does not result in "a size of the log file including log entries appears to be a substantially constant size". Also, fixed-size files do not have to be initialized with random data. A fixed-sized file such as that taught by the *IBM AIX* can be defined as a file that cannot exceed a certain size. For example, if the file is fixed at 100 bytes, the file can comprise any number of bytes from 0 to 100, but not exceed 100 bytes. The presently claimed invention, on the other hand, inserts random data into the log file at its creation so that the log file appears to be a substantially constant size no matter how many log entries are in the log file.

Accordingly, The Appellant asserts that it would not have been obvious to one or ordinary skill in the art at the time of the invention to insert random data into the log file when it

is initially created. A circular file can have a maximum size associated with it and when this maximum size is reached, old data is re-written, thereby creating a circular file. In fact, *IBM AIX* states that "the alog file…is a cyclic file so, when its size gets to the maximum, it is overwritten". This clearly shows that the circular file of *IBM AIX* is not populated with random data when it is generated, but is merely set to a maximum file size. In other words, the circular file of the *IBM AIX* is not populated with random data when it is generated to maintain a constant size, but is merely set to a maximum file size that can be reached.

Therefore, the inherency argued by the Examiner does not exist. If the presently claimed element is inherent and obvious, then the Appellant respectfully requests that the Examiner provide references teaching "…a log file of a relatively-fixed size which stores the log entry for the at least one set of data which has been encrypted, and wherein the log file includes the symmetric key which has been encrypted with the public key; random data in the log file when it is originally created and which is replaced by log entries so that a size of the log file including log entries appears to be a substantially constant size…" Accordingly, claim 1 (and similarly claims 10, 15, and 20) distinguishes over *Circenis*, the *OpenPGP standard*, and *IBM AIX* either alone and/or in combination with each other.

Independent claims 10, 15, and 20 recite similar to independent claim 1. Accordingly, the remarks and arguments made above with respect to independent claim 1 are also applicable to independent claims 10, 15, and 20. In view of the foregoing remarks and arguments the rejection of claims 1-22 should be reversed.

CONCLUSION

For the reasons stated above, the Appellant respectfully contends that each claim is patentable. Therefore, reversal of all rejections is courteously solicited.

Respectfully submitted,

Dated: January 15, 2009

By: /Jon A. Gibbons/
Jon A. Gibbons, Reg. No. 37,333

Attorney for Appellant

By: /Thomas S. Grzesik/
Thomas S. Grzesik, Reg. No. 54,139
Attorney for Appellant

Fleit, Gibbons, Gutman,
Bongini & Bianco PL
551 N.W. 77th Street, Suite 111
Boca Raton, FL 33487
Tel. (561) 989-9811
Fax (561) 989-9812

VIII. CLAIMS APPENDIX

1. A system that allows analysis of software running in a tamper- resistant environment, the system comprising:

a processor which monitors at least one instance of software execution, wherein the one instance is identified and selected by an end-user to be monitored by the processor, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, and wherein the processor creates a log entry with at least one set of data derived from the one instance of software execution in response to the one instance being identified and selected to be monitored, whereby the set of data is used to diagnose the software execution;

an encryption system which generates at least one symmetric key and encrypts the log entry for the at least one set of data using the symmetric key, wherein the encryption system encrypts the symmetric key using a public key associated with the encryption system;

a log file of a relatively-fixed size which stores the log entry for the at least one set of data which has been encrypted, and wherein the log file includes the symmetric key which has been encrypted with the public key;

random data in the log file when it is originally created and which is replaced by log entries so that a size of the log file including log entries appears to be a substantially constant size; and

a pointer which identifies a next storage location for a next log entry so that a last log entry can be determined and the next log entry can be positioned in a location in the log file after a previous log entry.

2. A system including the elements of Claim 1 wherein the system includes a transmission system for sending the log file, upon command, to a secure processing location away from the system in which the log file was created.

3. A system including the elements of Claim 1 wherein the system includes a system for wrapping around and filling the log file from a beginning when the log file has been filled, allowing the log file to remain at a substantially-constant size even after the log file has been filled with data and a new entry is received.

4. A system including the elements of Claim 1 wherein the system includes a mechanism for obscuring the log entry which has been created.

5. A system including the elements of Claim 4 wherein the mechanism for obscuring the log entry which has been created includes a printing function for writing into the log file.

6. A system including the elements of Claim 2 wherein the system includes a mechanism for receiving an indication from a user that transmission is desired and transmits the log file in response to that indication.

7. A system including the elements of Claim 1 wherein the system further includes a mechanism for receiving an input from an end-user that initiates logging of log entries into the log file each time logging is desired by the user.

8. A system including the elements of Claim 1 wherein the system further includes an initializing mechanism for determining each instance logging is to begin and initiating logging of log entries only in response to that initializing mechanism.

9. A system including the elements of Claim 1 wherein the system uses the public key to encrypt the log entry which has been created and a private key corresponding to the public key is used to decrypt the log entry which has been created at a secure location.

10. A method for diagnosing software in a tamper-resistant environment comprising the steps of:

generating a log file full of random data, wherein the log file is of a substantially constant size;

generating at least one symmetric key, wherein the symmetric key resides within the log file;

turning on logging and establishing a pointer for a location of a next logged software operation activity;

monitoring at least one software operation activity within the tamper-resistant environment and generating messages in response to at least one instance of software execution within the tamper-resistant environment, wherein the software operation activity is identified and selected by an end-user to be monitored, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user;

encrypting, in response to the monitoring, a record associated with each generated message using the symmetric key;

encrypting using the symmetric key, wherein a encryption system encrypts the symmetric key using a public key associated with the encryption system;

logging into a log entry into a log file at least one software operation activity relating to a generated message by replacing random data with an encrypted record of the software operation activity;

placing into the log file the symmetric key which has been encrypted with the public key;

moving the pointer when the log entry has been made to a next available log position;

wrapping the pointer to a beginning of the log file when the log file is full of log entries;

sending the log file to a secure location where the log file can be decrypted and analyzed; and

analyzing decrypted log file data and providing information for diagnosing software in the tamper-resistant environment.

11. A method including the steps of Claim 10 wherein the step of turning on logging includes the steps of receiving a user input indicating that logging is desired and initiating the logging in response thereto.

12. A method including the steps of Claim 10 wherein the step of at least one software operation activity further includes the steps of determining whether the software operation activity is to be logged, and if so, determining when to encrypt the software operation activity to obscure what is being logged.

13. A method including the steps of Claim 10 wherein the step of logging the software operation activity further includes the steps of determining a next available log position, replacing existing data in the next available log position with data from the software operation activity, and updating the pointer to provide a location of next logged software operation activity.

14. A method including the steps of Claim 10 and further including the step of receiving a command from a user that indicates that sending the log file to a remote location is desired and transmitting the log file in response thereto.

15. A method of analyzing the operation of software in a remote protected processing environment, the method including:

receiving from the remote protected processing environment an encrypted log file of substantially-constant size comprising at least one log entry with at least one set of data derived from at least one instance of software execution monitored in response to an end-user identifying and selecting the one instance of software execution, wherein the end-user is a user that initiates execution of the software at a system associated with the end-user, whereby the set of data is used to diagnose the software execution, wherein the set of data within the encrypted log file has been encrypted with at least one symmetric key included within the encrypted log file, and wherein the symmetric key has been encrypted by a public key associated with an encryption system;

determining a decrypting key for the encrypted log file and decrypting the encrypted log file;

determining a private decrypting key corresponding to the public key associated with the system;

analyzing, using the decrypting key and the private decrypting key, the log entry at the remote protected processing environment to determine whether an operation of the remote protected processing environment corresponding to the at least one set of data derived from at least one instance of software execution is appropriate; and

reporting results of the analyzing step.

16. A method providing the steps of Claim 15 and further including providing an instruction to initiate a logging of messages each time logging is desired by the end-user and an instruction to send to the encrypted log file to a remote system for analysis.

17. A method providing the steps of Claim 16 wherein the instruction to initiate logging of messages includes the step of initiating programming within the remote protected processing environment to replace information in the encrypted log file with encrypted information relating to the operation of the remote protected processing environment.

18. A method providing the steps of Claim 17 wherein the step of replacing information in the encrypted log file includes the step of replacing random data which was placed in the encrypted log file when it was created.

19. A method providing the steps of Claim 17 wherein the step of replacing information in the encrypted log file includes the step of using a pointer to a next location in the encrypted log file and the pointer wraps to a beginning of the log file after the encrypted log file has been filled.

20. A computer readable storage medium for analyzing software running in a tamper-resistant environment, the computer readable storage medium comprising instructions for:

recording at least one set of data serviced from at least one instance of software execution identified and selected by an end- user to be monitored whereby the set of data is used to diagnose the software execution wherein the end-user is a user that initiates execution of the software at a system associated with the end-user;

generating at least one symmetric key;

encrypting the symmetric key using a public key associated a client computer system

encrypting the recording of the at least one set of data using the symmetric key;

recording the at least one set of data, which has been encrypted sequentially in a storage block of a substantially fixed size, wherein the storage block includes the symmetric key which has been encrypted with the public key;

maintaining a pointer to a next available location for recording the at least one set of data sequentially in the storage block;

responding to a command and sending an encrypted log file comprising the at least one set of data which has been encrypted and sequentially recoded in the storage block to a remote location for decryption and analysis.

21. The computer readable storage medium of claim 20, further comprising instructions for:

initializing the storage block of a substantially fixed size with random information which has been encrypted to provide a block of apparent data.

22. The computer readable storage medium of claim 20, further comprising instructions for:

writing the at least one set of data which has been encrypted and recorded in a sequential order in the storage block of the substantially fixed size and for wrapping around when an end of the storage block of the substantially fixed size is reached.

IX. EVIDENCE APPENDIX

NONE

X. RELATED PROCEEDINGS APPENDIX

NONE